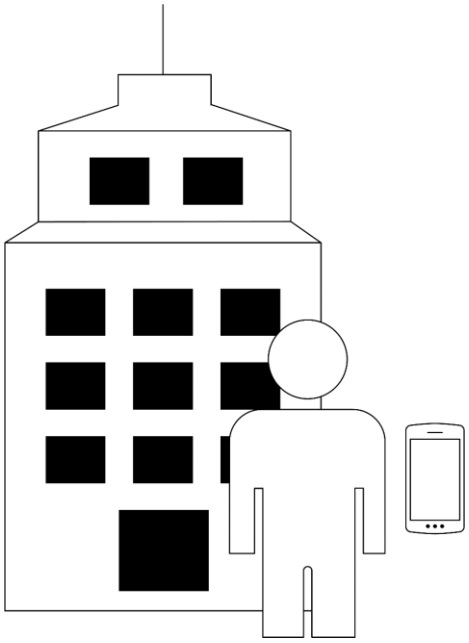

Navegação "Indoor" baseada em "fingerprinting" de redes sem fios e em informação sensorial de smartphones

Miguel Ângelo Ferreira de Madureira
Mestrado Integrado de Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2015

Orientador
Sérgio Crisóstomo, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto

Coorientador
Rui Prior, Professor Auxiliar
Faculdade de Ciências da Universidade do Porto

U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Abstract

Positioning and navigation have always been concerns of humanity. The tools we've built in support of these concerns are numerous, and by and large they do not work indoors. The growing use of smartphones, pocket computers, has afforded us an opportunity to extend navigation and positioning indoors, to provide services like finding out where an office is, or information about points of interest.

This has motivated us to develop two independent positioning systems which we successfully deployed them on an Android smartphone, using a server in support of it. Described in this work are the positioning systems: a dead reckoning inertial navigation system using a pedometer - making use of the smartphone's accelerometers - and a compass and a Wi-Fi fingerprinting system that matches observed Wi-Fi access point signal strengths to a radio map.

Keywords: Inertial Navigation System, Wi-Fi, Fingerprinting, Android, accelerometer

Resumo

Navegação e posicionamento foram desde sempre preocupações da humanidade. No entanto, as numerosas ferramentas que construímos para estes fins não funcionam, geralmente, dentro de portas. Com o uso cada vez maior de *smartphones*, computadores de bolso, surgem oportunidades para colmatar esta falha, dando azo a serviços tais como descobrir onde fica um escritório, ou ver informação sobre pontos de interesse.

Esta situação motivou o desenvolvimento de dois sistemas independentes de posicionamento, implementadas sobre um *smartphone* Android, com o auxílio de um servidor. Este trabalho descreve ambos os sistemas: um sistema de navegação inercial, que usa um pedómetro - criado sobre os acelerómetros de um smartphone - e uma bússola, e um sistema de 'fingerprinting' Wi-Fi, que relaciona observações feitas a pontos de acesso Wi-Fi com um mapa de rádio.

Palavras-chave: Sistema de navegação inercial, Wi-Fi, Fingerprinting, Android, acelerómetro

Contents

List of Tables

List of Figures

Acronyms

AGPS Assisted GPS

ADOA Angle Difference of Arrival

AOA Angle of Arrival

AP Access Point

API Application Programming Interface

BSSID Basic service set identification

CPU Central processing unit

DBMS DataBase Management System

DR Dead Reckoning

GNSS Global Navigation Satellite Systems

GLONASS Globalnaya Navigatsionnaya Sputnikovaya Sistema

GPS Global Positioning System

HTTP HyperText Transfer Protocol

IEEE Institute of Electrical and Electronic Engineers

INS Inertial Navigation System

IR Infra-Red

ISM Industrial, Scientific and Medical radio band

JSON JavaScript Object Notation

LAN Local-Area Network

LOS Line-Of-Sight

MEMS Micro-electromechanical systems

NNSS Nearest-neighbour in signal space

NNSS-AVG Nearest-neighbour in signal space, averaged

LPD433 Low-powered device 433MHz

PAN Personal-Area Network

RF Radio-Frequency

RFID Radio-Frequency Identification

RSSI Received Signal Strength Indicator

SNR Signal-to-noise ratio

SSID Service set identification

TDOA Time Difference of Arrival

TOA Time of Arrival

Chapter 1

Introduction

Positioning and **navigation**, knowing where we are and which way to get to somewhere, have been needs of ours for millennia. These needs have motivated us to find solutions and build tools to assist us in locating ourselves on the surface of the Earth. The earliest navigators were aided by the motion of stars. In the Medieval Era, the astrolabe, a tool that determines latitude in a marine setting, was first built and used. During the Age of Discovery, the first accurate timekeeping devices were made, enabling us to precisely determine longitude; more recently and currently the culmination of our efforts, during the latter half of the 20th and early 21st century, the deployment of satellite constellations - Global Positioning System (GPS), Globalnaya Navigatsionnaya Sputnikovaya Sistema (GLONASS), Galileo, collectively known as GNSS - gave us fast and accurate positioning and navigation, for military, commercial and personal use.

Since the latter half of the 20th century, there have been enormous strides taken towards the miniaturisation of consumer electronics. In particular, the advent and proliferation of affordable, internet-enabled, handheld computers, commonly known as smartphones, has brought some or all of these GNSS technologies into the hands of consumers, who make daily use of them. Because smartphones are ubiquitous and heavily used, the limitations of GNSS become readily apparent. The one that most interests us in this instance, none of these systems are accurate or usable indoors - these being radio-based systems, not having Line-Of-Sight (LOS) to the satellite constellation as well as fading imposed by a building's walls and multipath propagation ensure that the signal either becomes too weak to interpret or accuracy is compromised. Because of this, users are forced to study an unfamiliar building's layout through a map, and a need to extend navigation and positioning systems indoors has arisen, and with it, a new set of problems to

solve.

The high availability of smartphones has provided the means to address the indoor navigation and positioning problem. Because they commonly include a vast array of sensors, such as Wi-Fi, Bluetooth and cellular radios, inertial sensors such as 3-axes accelerometers and gyroscopes, magnetic sensors and barometers, a vast array of avenues to explore are opened. Because users are already used to GNSS, a solution that extends the GNSS analogy indoors, capable of locating a user and directing them towards their destination, would be both more convenient and more functional than a static map.

1.1 Objectives

The ultimate goal is to develop an indoor positioning system that is usable in a smartphone.

First, the application shall collect data from the inertial sensors and magnetometers and store it for future processing.

Second, the application shall collect data from the Wi-Fi radio sensor, and store it for remote processing.

Third, and in support of these goals, an application that shall run on a server machine will perform the computations that the smartphone cannot.

The fourth and final objective shall be the melding of these disparate data into a model that is representative of the physical position of the user.

1.2 Design Overview

The proposed solution is divided into two parts: a program running on the smartphone, that collects data from two sources: the inertial sensors, namely the compass and the accelerometer, and the Wi-Fi adapter, which produces Wi-Fi scans periodically. The inertial sensor samples are consumed on the smartphone itself, used in the pedometer algorithm. This algorithm produces a displacement - a step taken - and the direction in which the step was taken. The scans are sent to a remote machine, running a pattern matching program. This program emits a set of likely positions where that scan was performed, and sends this set back to the smartphone.

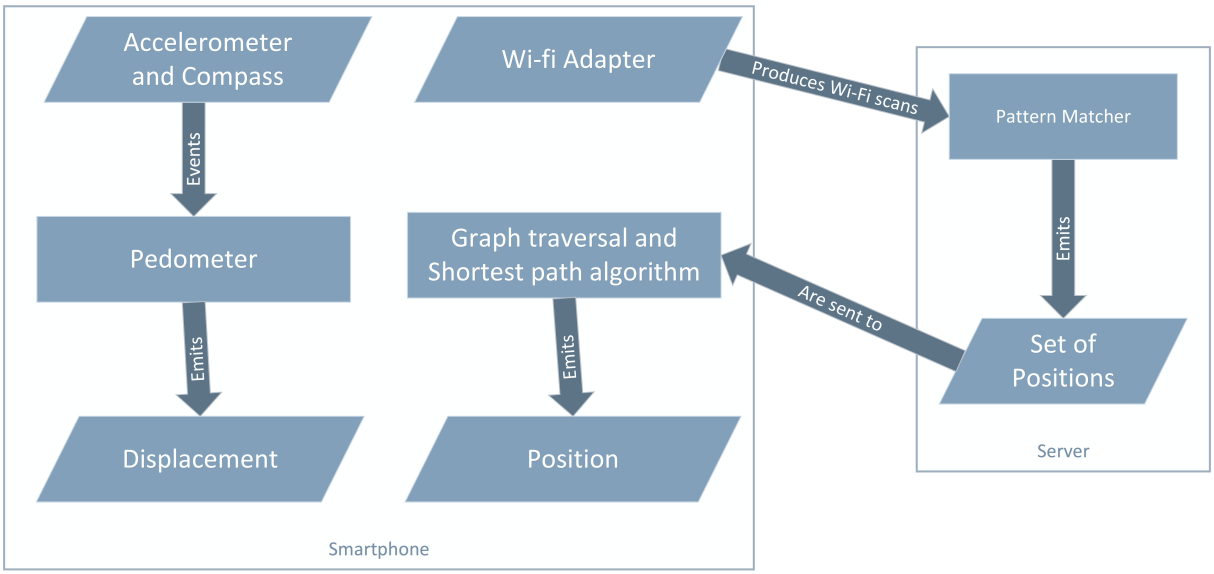

Figure 1.1: Diagram detailing the high-level data sources and processes of the proposed solution.

1.3 Structure

This dissertation is laid out thus: Chapter 2 describes the various existing solutions that address the problem. Chapter 3 details the Dead Reckoning system, its implementation, and problems encountered and solved. Chapter 4 details the Location Fingerprinting system, its implementation, problems encountered and solved.

Chapter 2

Background

The commonplace use of smartphones and GNSS shows that there is a great demand for positioning solutions. Their use indoors comes as a natural demand, which hasn't quite yet been fulfilled. Furthermore, commercial interests have started focusing their attentions towards this area, either by targeting information towards users based on their location, be it ads or the displaying of information relevant to the user's position, or by enabling the collection of statistics about users' habits [1].

Research efforts into this area are numerous, focusing mostly on the core challenge of this problem: an accurate positioning technique that works indoors. This section describes various positioning technologies and techniques that have been researched and considered for the project.

2.1 Positioning Technologies and Techniques

Through the ages, Man has found its way through the planet. The ways in which this is achieved have evolved with time, and taking stock of them is important. In this section, we detail the techniques relevant to the indoor positioning problem as they arise in the historical timeline, along with their feasibility.

2.1.1 Dead reckoning and Inertial Navigation

Dead Reckoning (DR) is possibly the oldest method for positioning. This technique consists in the estimation of an object's position by calculating the distance and direction travelled relative

to an earlier position as opposed to a fixed landmark or other form of absolute positioning.

DR is commonly used for marine navigation and in non-assisted aviation. Because it does not take into account disturbances of the medium of travel, such as water or air currents, this technique is subjected to cumulative errors. Without the aid of other positioning systems or a reference point, these errors grow unbounded.

An Inertial Navigation System (INS) is an extension of this idea, that makes use of inertial sensors, such as linear accelerometers, speedometers, gyroscopes and barometers to continuously update the position of an object.

It is possible, for instance, to build a pedometer that can estimate the distance travelled and the direction in which a step was taken, using only the accelerometers and a compass.

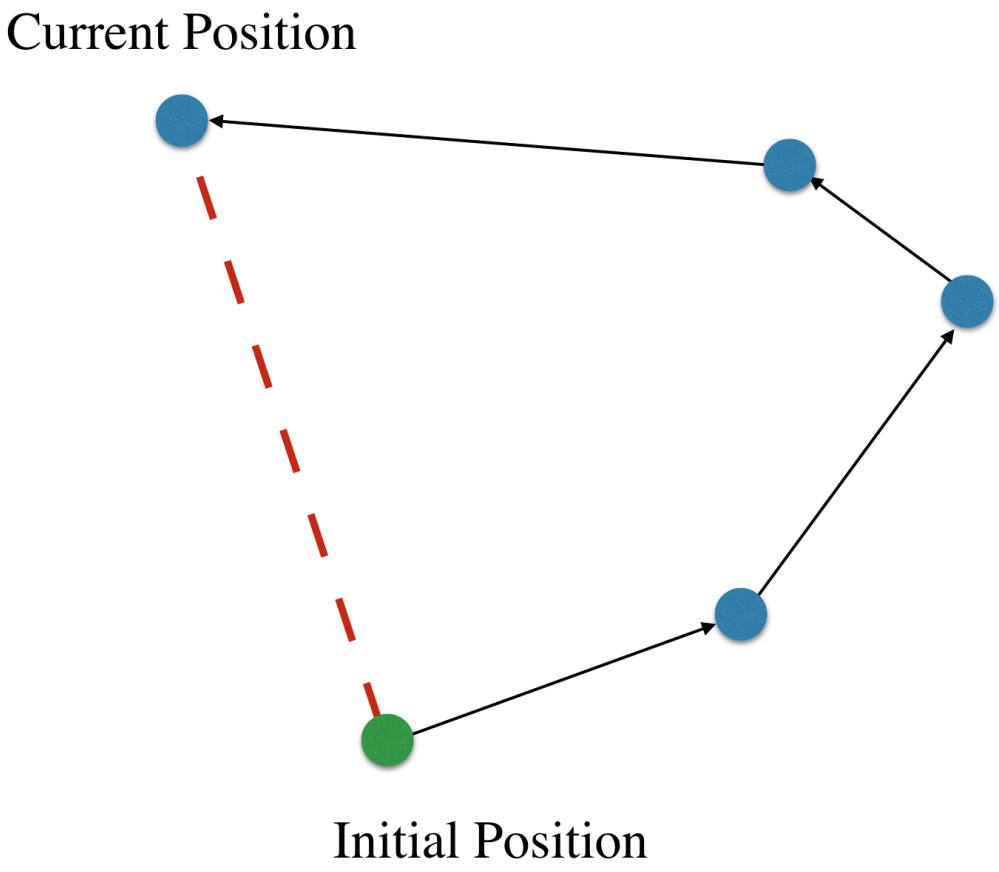

Figure 2.1: Dead reckoning example. From an initial position, given a direction and speed, it is possible to estimate subsequent positions. This technique is subjected to unbounded errors, since any error in any estimate will propagate.

2.1.2 Global Satellite Navigation Systems

Satellite navigation represents today the most advanced positioning system implemented. Relying upon satellites in orbit of the Earth, these systems enable objects equipped with radio receivers the accurate - to an error in the order of metres - and absolute positioning anywhere on the planet's surface, provided there is LOS to a number of satellites. The currently deployed global systems, termed GNSS include the American GPS, the Russian Federation's GLONASS and the European Union's Galileo.

These systems require a LOS from the receiver to the satellites - usually four or more - because any degradation in the signal, such as multipath propagation - weakening the signal strength, or destructively interfering with the signal that took the direct path - and fading imposed by obstacles - such as walls which, depending on material, may be completely opaque to the signal - renders it unusable. Therefore, these systems are not directly usable indoors. Even if the user is located outdoors but in an urban setting, the signal from these satellites degrades, owed to multipath propagation, fading and occlusion due to buildings, walls and other obstacles. This imposes a severe degradation of accuracy, assuming localisation is indeed possible. To alleviate this issue, one solution has been deployed.

Assisted GPS (AGPS), primarily used in cellular devices such as smartphones, improves upon GPS' performance by adding information about the conditions of the cellular network the device is connected to. This is done in two ways: the cell tower keeps information about the overhead satellites, reducing the time required for a lock by the mobile device, or by offloading partial GPS data to a remote server with plentiful computational power and knowledge of the radio conditions local to the user. Unfortunately, there is a too-severe accuracy penalty when using this method indoors, for the same reasons as basic GPS.

Pseudo-satellites, also known as Pseudolite [2], are indoor-installed systems that extend GNSS coverage. Highly accurate and transparent to existing GNSS terminals, their main drawback is the high cost and need of specialized hardware.

2.1.3 Location Fingerprinting

Location fingerprinting techniques attempt to address the shortcomings of GNSS use indoors by making use of hardware that is commonly used in this setting. This hardware, being much closer

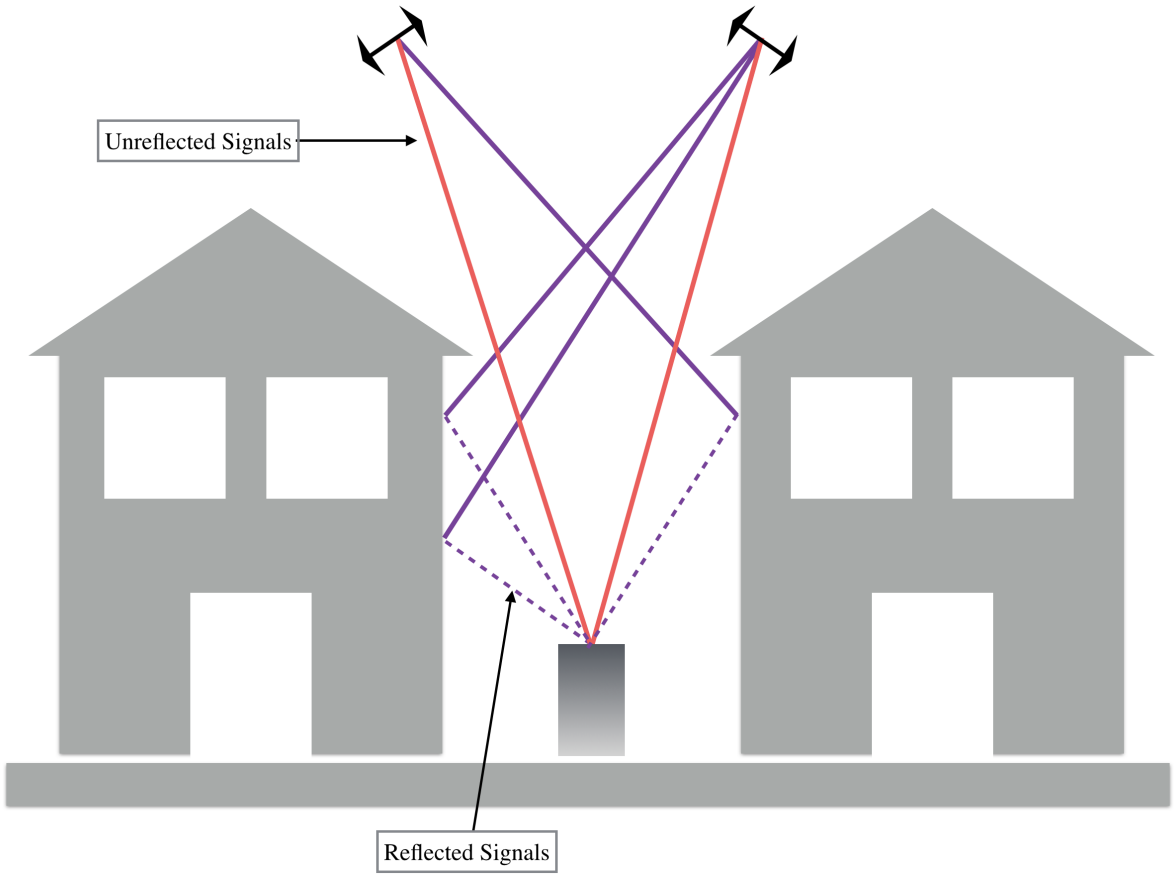

Figure 2.2: In an urban canyon, a GNSS receiver will have its view of the satellites obstructed, limited to those almost directly overhead. However, reflected signals from these and overhead satellites will interfere with direct signals, degrading the estimated position.

to an object attempting to find out its position, provides much stronger signal strengths and ubiquitous coverage.

This fact allows for the use of already-existing sensors that are designed to use this hardware, such as Wi-Fi and Bluetooth hardware adapters. Maps - or fingerprints - are then created, which associate a location in physical space to some aspect of the radio signal, such as the Received Signal Strength Indicator (RSSI), Basic service set identification (BSSID) or Service set identification (SSID), or a combination of these, in the case of Wi-Fi.

Location fingerprinting algorithms commonly proceed in two distinct phases: an *offline* or *training* phase, where signal strength measurements are collected, and an *online* phase, where the location of an object is determined.

2.1.3.1 Wi-Fi

Wi-Fi is a wireless technology for Local-Area Network (LAN). Operating on the 2.4GHz and 5Ghz ISM bands, Wi-Fi products comply to the Institute of Electrical and Electronic Engineers (IEEE)' 802.11 standards. The use of this technology is common and widespread. Buildings ranging from shopping centres, public or government offices, university campuses, apartment buildings or town houses have at least one Wi-Fi access point. Because of this, solutions based on this technology have a lower cost of implementation.

2.1.3.2 Bluetooth

Bluetooth is a wireless technology for Personal-Area Network (PAN). Operating on the 2.4GHz Industrial, Scientific and Medical radio band (ISM) band, Bluetooth was standardized in the IEEE's 802.15 standard.

Because of the low power typically used in Bluetooth applications, the use of this technology would require a large number of beacons to cover a large area, which could be prohibitively expensive. However, there applications of Bluetooth technologies in use for very coarse localisation - where the accurate position is not required - in museums, malls and sports arenas [1].

2.1.4 Techniques

Owing to Wi-Fi's ubiquitousness and low cost, heavy focus was given to positioning techniques that make use of it. This section details some of the existing techniques that have been researched and considered.

2.1.4.1 RADAR

RADAR [3, 4] is a user location and tracking system built on top of a Wi-Fi infrastructure, deployed inside an office building. In their original experiment, objects were tracked by having them broadcast packets periodically and having the base stations observe the signal strength associated with them. This signal strength is then compared to a map, that pairs signal strength with a physical location. The closest match is then guessed to be the actual object's location.

2.1.4.2 NIBBLE

This fingerprinting technique models the environment as a Bayesian Network, and offers opportunities for user profiling. [5]

Using a set of locations L, set of observed values E, and R sensor readings, it answers the question $p(L) =?$ by constructing the network $p(E|L)$. $p(L|R)$ is the *a posterior* probability distribution of L, the probability that we are in a location v_i given R.

This method also tries to address various sampling conditions, by including information about the observed Signal-to-noise ratio (SNR) in E. By taking multiple samples at multiple times of day and number of persons around, this method addresses the impact of persons on the RF environment.

The $p(L)$ *a priori* distribution can be constructed by directly sampling the environment in an offline phase. This, however, is not required. If not done, then the user is expected to provide their own mappings.

2.1.4.3 COMPASS

This method [6] addresses the impact of the user in the radio environment by pairing orientation information (using a compass) with the radio map. Only similarly oriented patterns are selected when considering a position, thereby constraining the user's position greatly.

2.2 Devices

2.2.1 Smartphones

Smartphones are portable computers capable of running an operating system, that can make phone calls. Because of this, arbitrary programs may be executed on them, such as navigation applications that make use of GNSS, calendar or video games. They are commonly internet-enabled, either via Wi-Fi or cellular internet.

Commonly, smartphones come equipped with a full suite of Micro-electromechanical systems (MEMS), additional sensors such as accelerometers, gyroscopes, compasses or barometers. Because of this, a world of motion-enabled applications is possible, taking advantage of these computers' mobility and ubiquitous use - 56% of Americans use these devices [7].

2.2.2 Micro-electromechanical systems

MEMS are essentially very small machines, ranging from the micro- to the millimetre scale, that may contain moving parts. These machines fall into one of two categories: sensors and actuators. Both of these categories convert energy from one form into another, usually mechanical action into electrical signals.

Over the past few decades, these sensors have been built to cover almost every possible use, ranging from sensors for temperature, pressure, inertial forces, magnetic fields, among others, and are integrated into commonplace devices, such as smartphones.

Due to this, smartphones are good candidates for the use of dead reckoning.

2.2.3 Smart Watches

Smart watches are wrist-wearable computers that can be paired with smartphones in order to exchange data, such as displaying information about unread messages, the weather or the time of day.

These devices are relevant for our application because they may be an additional data source, by compiling information from a single user using two different devices.

2.3 Existing Applications

The following is a list of existing smartphone applications that purport doing indoor positioning.

2.3.1 Concordia University Indoor Positioning System (IPS)

Authored by Richard Btaiche, Eldar Khasmamedov, Elie Milan, Nikolaos Papadakis, this application is available on the Google Play store under the title "Indoor Positioning System" [8]. This application's goal is to provide an indoor positioning and navigation solution for the Concordia University's campus. Its features include:

- Lists points of interest.
- Ability to share with other users one's location.
- Path finding.
- Uses inertial sensors to aid localization.
- Augmented reality - through the use of the device's camera, show information about classrooms, etc.

The method for localization is unknown, although it is stated that it uses triangulation and proximity to Wi-Fi access points.

2.3.2 Projet Wifi - Positionnement Interieur

Authored by Bao-Ngoc N'Guyen Van, the application purports to fingerprint the radio environment, mapping these fingerprint to grid locations [9]. The fingerprinting method is unknown. However, interacting with the application revealed that the grid locations store the RSSI of the Wi-Fi APs selected for sampling.

2.3.3 Wifarer

This application [10] only works in select partner venues; at all other locations, it works as a typical GPS app, plotting the user's location on a Google-provided map.

The application does not make use of any inertial, magnetic or barometric sensor, using only the Wi-Fi and Bluetooth radios.

2.3.4 SmartNavi

Authored by Christian Henke, this application is available on the Google Play store [11]. It proposes the augmentation of GPS navigation by using the inertial sensors available on the device to position the user relative to the last available GPS position. Its features include:

- Overlays the user's position on a Google/OpenStreetMaps-provided map.
- Allows for GPS-assisted inertial drift correction.
- Allows exporting of the recorded path taken by the user.
- Allows for offline storage of maps, enabling a totally offline positioning and navigation experience.
- Open source.

The method for sensor fusion is unknown at this time. The distance is calculated by estimating step length from the user's height via an unknown conversion.

2.3.5 Comparison

The following table compares the existing applications as they relate to positioning technique. It is interesting to note that most of these applications are only available for Google's Android Operating System, possibly due to API restrictions or just market share - Android accounts for 82.8% of the market [12].

Table 2.1: Comparison of different indoor positioning applications for smartphones

	Inertial Navigation	Fingerprinting	Operating System	Source
IPS	Yes	Yes	Android, unknown version	Closed
Projet Wifi	No	Yes	Android >=2.1	Closed
Wifarer	No	Yes	Android >=2.3, iOS >= 7	Closed
SmartNavi	Yes	Yes	Android >= 2.3	Open

Chapter 3

Inertial Navigation System

In this chapter we discuss in detail the inertial navigation solution, with an overview of the problems posed and solved.

3.1 Defining the problem

A smartphone-mounted sensor suite, composed of MEMS, may find itself in any number of orientations. Consider the typical use case for a smartphone - it sits either in one's pocket, is held by one's ear when talking through it, or is held in front of the user, with the screen in view. In order to not constrain the user while they are using a dead reckoning system, it should be able to function in all these scenarios. The device's orientation should, ideally, not be a factor that determines whether the system works at all, nor a factor in its accuracy.

The problem is made more complex by the freedom allowed the mechanism. It must be able to tell apart between spurious motions - random jitters, including sensor noise, a motion of the user's arm, a user turning - from an actual motion of the body in a direction.

Therefore it is necessary to find a solution that minimizes or avoids these problems, while providing the highest possible degree of accuracy for an inertial system.

3.2 Pedometer - compass algorithm

A typical Android smartphone has available to it a few inertial sensors, namely linear accelerometers magnetic field sensors.

Google encourages the use of these sensors to determine the device's position [13], particularly, the devices' rotation relative to the Earth [14, 15]. Because of the problems mentioned in 3.1, particular the random jitters a smartphone may be subjected to, we avoided making use of the raw sensor data, opting to construct an indirect solution.

The solution we considered most unambiguous was that of a step counter, or pedometer. Because a step induces a very large vertical acceleration upon the human body - evident in figure 3.1, larger than that along any other axis, it was considered the better candidate that would solve both problems at once.

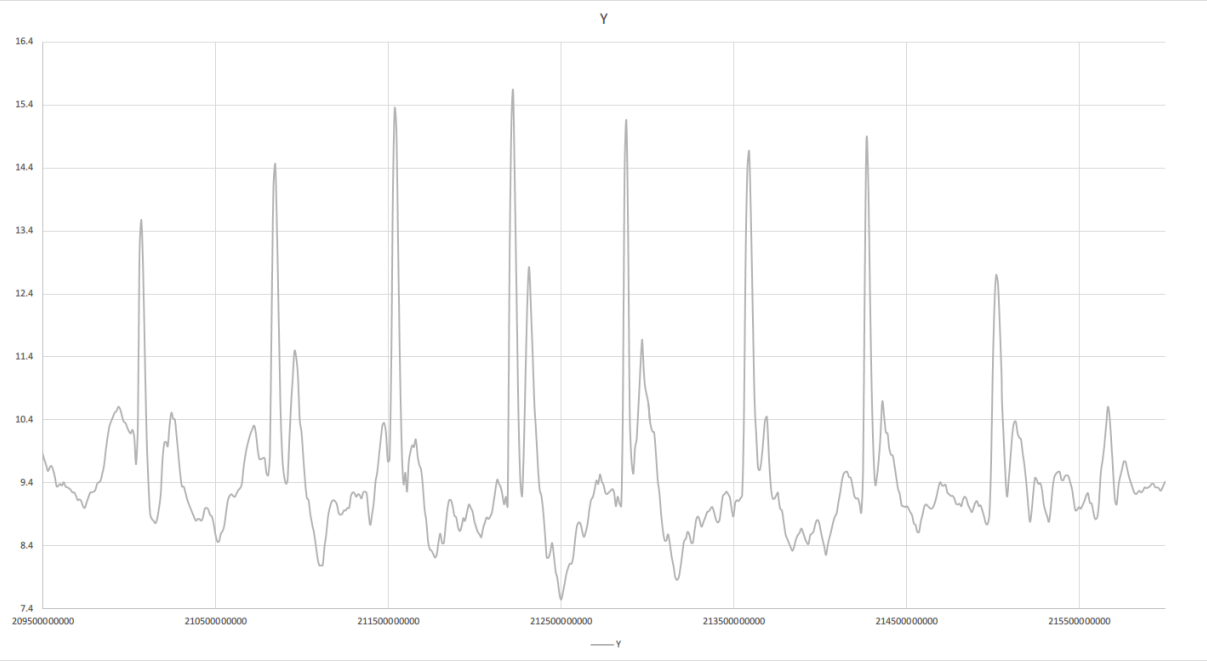

Figure 3.1: Raw accelerometer data collected as a subject was walking, using a shoulder-mounted smartphone. The tallest peaks are instants where steps occurred. Readings are taken from the axis that is aligned with gravity. X axis units are time in nanoseconds since the smartphone was turned on and Y axis units are metres per second.

In order to know the direction in which a step was taken, the rotation around the device's Z axis, as seen in figure 3.2, which can be interpreted as a compass, with the angle of rotation

representing an angle relative to the magnetic north.

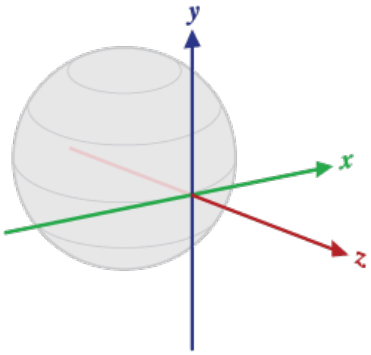

Figure 3.2: An Android smartphone's standard axis definition relative to the world. The globe represents the Earth. The X axis is tangential to the ground and points East; Y axis is tangential to the ground and points North; Z axis is perpendicular to the ground and points towards the zenith. Image ©Google, Available under Creative Commons Attribution 2.5 Generic.

3.3 The Pedometer algorithm

Neil Zhao's algorithm [16] for a simple accelerometer-based pedometer was chosen because it addresses all the concerns raised in 3.1, meets all the requisites laid out in 3.2 and is simple to understand and implement.

3.3.1 Algorithm description

Basic assumptions The algorithm as originally laid out assumes a constant and fixed sampling rate of 50Hz, and used an ADXL345 accelerometer as its data source. Enough memory must be available to store the maximum and minimum accelerometer values for all three axes, along with three variables that represent the current accelerometer sample - **current_sample**, and two samples that will be important for the decision of whether a step has been taken - **sample_new** and **sample_old**.

Data smoothing Accelerometer samples are averaged over the last four samples, in order to make the data smoother.

Dynamic Threshold The algorithm keeps in memory the maximum and minimum value of all the accelerometer's axes, continuously updated. The average value of these is called the **dynamic threshold**, and is updated every second. This threshold value is used to decide whether a step has been taken.

Dynamic Precision A user set parameter, this precision value determines the smallest change in acceleration that must occur before the algorithm proceeds.

Zhao's algorithm then proceeds as follows:

- Determine the axis to listen on. This is done by calculating which axis showed the greatest change in acceleration since the last sample.
- Unconditionally move the value stored in **sample_new** to **sample_old**.
- If the change in acceleration in that axis is larger than the user-determined **precision**, move **sample_value** to **sample_new**.
- If **sample_new** < **sample_old** and **sample_new** < **dynamic threshold** then a step is counted.
- If this step is part of a rhythmic pattern, then it is considered valid. If not, it is considered a jitter and is discarded.

Zhao also outlines a simple formula for the displacement undertaken by a person:

$$Distance = number\ of\ steps \times distance\ per\ step \tag{3.1}$$

With a distance per step, determined experimentally by Zhao, as follows:

Table 3.1: Stride length as a function of speed and user height

Steps per 2 seconds	**Stride (meters per second)**
0-2	Height / 5
2-3	Height / 4
3-4	Height/3
5-6	Height/2
6-8	Height
>= 8	1.2 x Height

3.3.2 Implementation

Although very promising and easy to implement, the algorithm presented a few problems when implemented on an Android Smartphone. The following changes were therefore effected, to ensure a correct performance.

Making it sampling-rate independent The original algorithm assumed a constant sampling rate, which isn't a given on a pre-emptive multitasking operating system like Android. Depending on the particular smartphone, the sampling rate might vary between 50Hz and 10Hz, depending on the hardware, operating system version, and Central processing unit (CPU) load. Because of this, the algorithm was first altered to become as sampling-rate independent as possible - instead of presuming a fixed sampling rate, the samples' timestamps are recorded and a set of them is traversed using time as an index.

This was achieved by creating variable length circular buffers. As each accelerometer sample arrives, it is inserted into the buffer and its timestamp is evaluated to check how much time has passed since the oldest element was inserted. The update process then proceeds as detailed in 3.3.

Rythmic pattern Although a suggested mechanism, a concrete description of it is not available in the original description of the algorithm. We implemented a simple mechanism, which proceeds as follows:

- Initially, no pattern exists.

- When a step is detected, its timestamp is recorded.
- When another step occurs, the time difference is recorded.
- If subsequent steps occur within the interval $[difference - 40\%\ ,\ difference + 40\%]$, all 3 steps are considered valid and counted. Further valid steps are counted.
- If a step falls outside the interval, all steps are discarded, and the pattern is reset.

Oversensitivity Possibly due to the fact that the algorithm was not originally designed to work with a handheld device, it is overly sensitive, and detects steps when none occurred. We observed that, whether shoulder-mounted or handheld, there was a too high false-positive rate - around 70% of detected steps were false. The false positive problem is illustrated by figures 3.3 and 3.4.

The problem is due to the fact that the step detection condition is not strict enough. Any curve segment that had a negative slope which had a sample exceeding the **precision** argument and occurred below the **threshold** triggered a step.

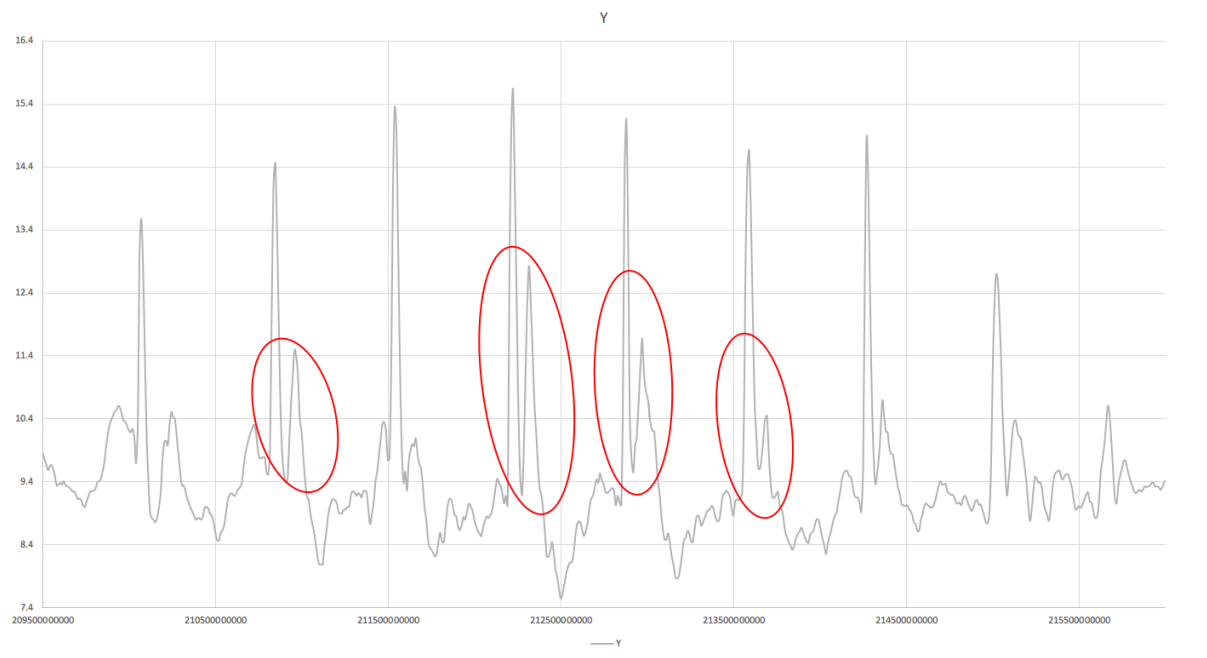

Figure 3.3: Same data-set from figure 3.1. Circled in red are instances in which a two-step sequence was detected.

Compounding the issue was the step pattern mechanism. In a situation like figure 3.3, each two step pattern would prime the pattern matcher, which would discard both because the next

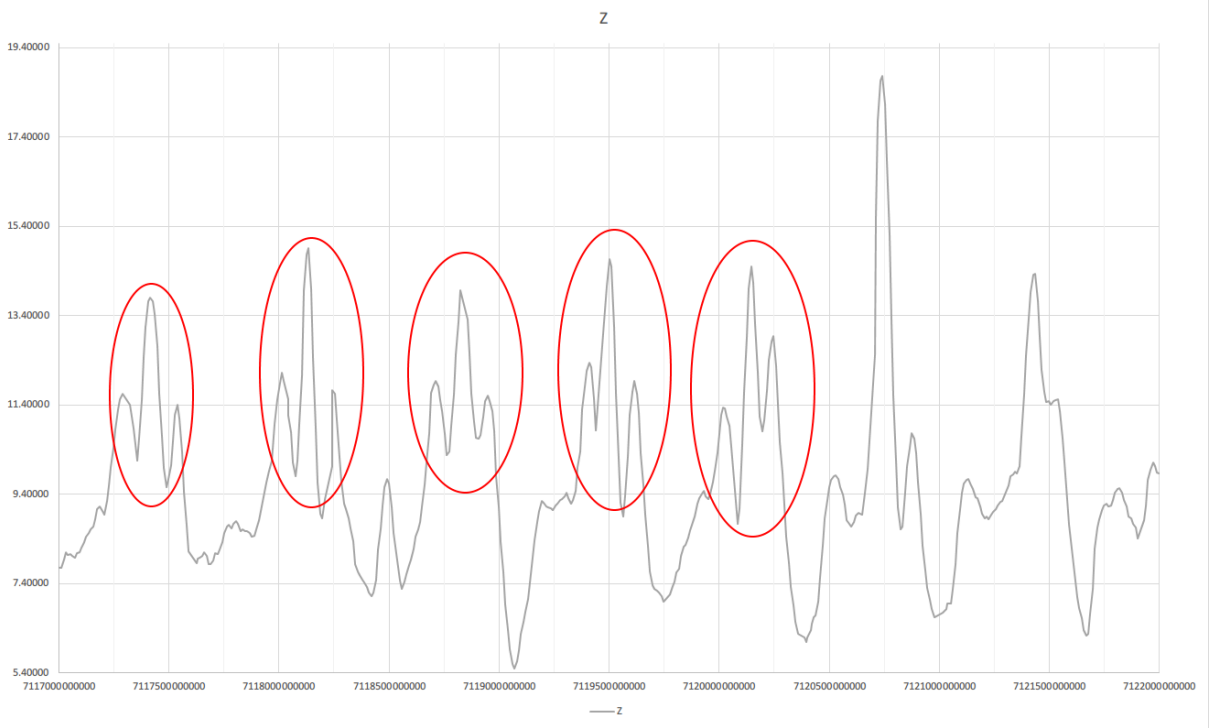

Figure 3.4: Accelerometer data from a hand-held smartphone. Circled in red are instances in which a three-step sequence was detected.

step occured outside the interval. More seriously, in a situation like figure 3.4, each triple step set would be counted as three valid steps, where only one step occurred.

By observing that steps always occur at the moments of highest acceleration, the solution presented itself: force the **threshold** to be crossed on the positive slope segment of the curve, by the **precision** amount. While it has not, no steps may occur. This solved the false-positive problem.

Saving step information Due to the need to fuse steps with direction, the timestamps for steps are recorded, so that they may be matched with a direction.

3.4 Compass

Because Zhao's solution only produces a displacement, it was necessary to implement a compass so that the system can emit a prediction of a direction along with the displacement. This adds a constraint to the system. Rotation around the device's Z axis, when expressed through world

co-ordinates, gives us the angle relative to the magnetic North - therefore, the device must be held in front of the user with the device's Y axis roughly laying along the direction of travel. This was deemed an acceptable compromise given the gain.

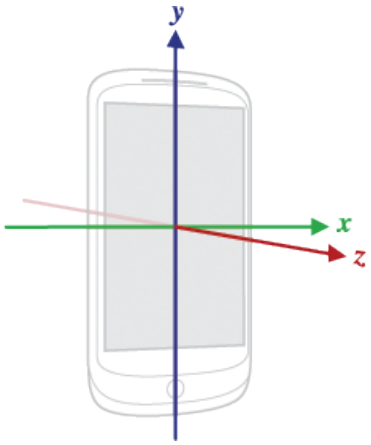

Figure 3.5: An Android device's co-ordinate system. The Y axis must coincide with the direction of travel in order to know the accurate direction of travel relative to the magnetic north. Image ©Google, Available under Creative Commons Attribution 2.5 Generic.

Therefore, the pedometer algorithm was augmented with a compass so that it works as intended in the envisioned typical use case - being held by the user, with its screen facing them.

The compass was implemented using the smartphone's built-in magnetometer, and Android OS's interpretation of the data given by both the accelerometer and magnetometer [15]. Simply, using data from these two sources and the Android Application Programming Interface (API), the angle to the magnetic North may be known at all times.

This allows for the construction of a rotation matrix R

$$R = \begin{bmatrix} E_x & E_y & E_z \\ N_x & N_y & N_z \\ G_x & G_y & G_z \end{bmatrix} \tag{3.2}$$

where x, y, z are the axes as illustrated in figure 3.2 and

$$E = (E_x, E_y, E_z) \text{ unit vector that points East}$$
$$N = (N_x, N_y, N_z) \text{ unit vector that points North}$$
$$G = (G_x, G_y, G_z) \text{ unit vector that points opposite of gravity}$$

We extract the angle relative to north, or the **azimuth** ϕ with [17]

$$\phi = \arctan\left(\frac{E_y - N_x}{E_x + N_y}\right) \tag{3.3}$$

When computed, this rotation is stored along with the time the data that led to its computation was generated. This allows a rotation to be matched with a step.

3.5 Fusing the pedometer with the compass

When a step is detected, the system looks at its timestamp and searches the compass historic data to find the direction that was computed closest to when the step occurred. Finally, with these two data, we can perform dead reckoning, as illustrated by figure 3.6.

3.6 Summary

In this chapter we described the two major components of the inertial navigation system: the pedometer algorithm, by Neil Zhao and alterations required to it when implemented in an Android smartphone; then, the compass, which enables the system to know in which direction it moved; and finally, the way the two are fused together, to emit a new position.

3.7 Inertial Navigation System Overview

This section presents a high level overview of the inertial navigation system thus produced.

Figure 3.6: Dead reckoning system exemplified. On screen is the displacement recorded over time.

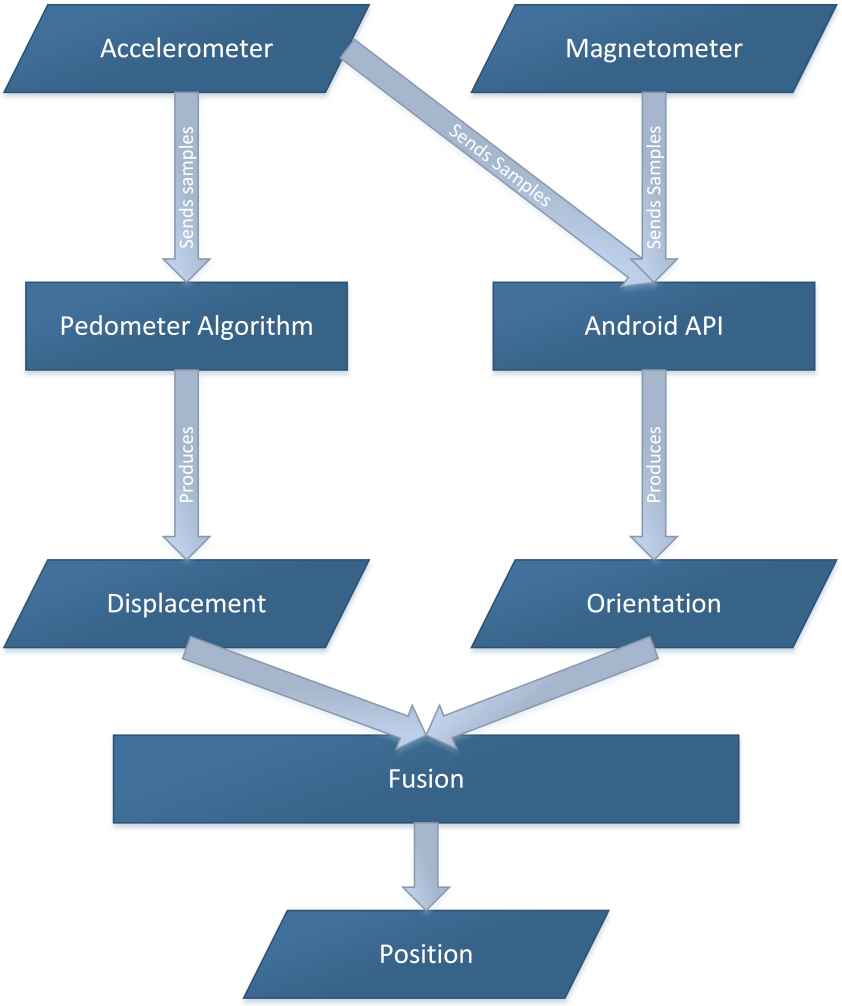

Figure 3.7: Overview of the inertial navigation system.

Chapter 4

Location Fingerprinting

In this chapter we discuss in detail the RADAR[3, 4] algorithm and the details of its implementation. The first section addresses the RADAR algorithm proper, detailing its workings, and the last section details the actual implementation of RADAR.

4.1 RADAR

Locating a user who is moving indoors without using a system whose sole purpose is location is the problem that the RADAR algorithm proposes to solve. As opposed to purpose-built user tracking systems, such as those using Radio-Frequency Identification (RFID) tags or Infra-Red (IR) networks, RADAR addresses the problem by making use of Radio-Frequency (RF) local-area wireless networks, i.e. Wi-Fi, which enjoy the benefits of ubiquitousness, range, scalability, and ease of maintenance.

The RADAR algorithm was described in two different papers: the first paper [3] detailed the original implementation, and the second[4] improvements upon the original. The following sections will consider the entirety of both papers as a single description.

4.1.1 Experimental Testbeds

Two different experimental testbeds were used, both indoors. The first testbed was deployed on the second of a three-floor building, which had more than 50 rooms. The layout is visible on figure 4.1. Three Wi-Fi access points were placed on this floor, covering it in its entirety with a

Wi-Fi signal. The mobile user was represented by a Pentium-based laptop running the Windows 95 operating system.

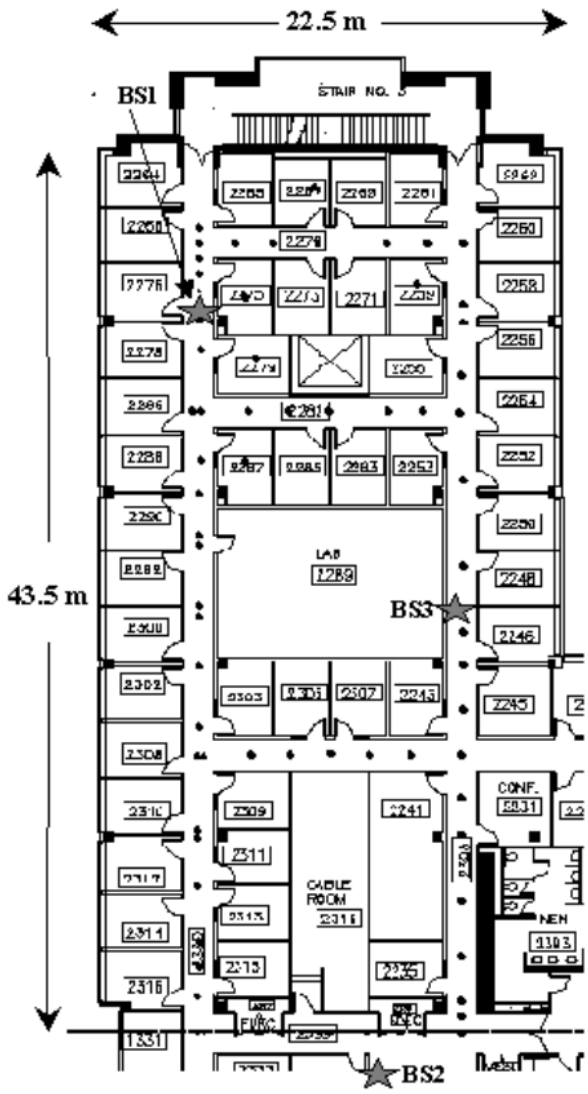

Figure 4.1: Floorplan of the site chosen for the first testbed. Stars represent the locations of the Wi-Fi APs. Dots represent the locations where signal strength samples were recorded. Image by Bahl, Paramvir and Padmanabhan, Venkata, RADAR: An In-Building RF-based User Location and Tracking System, 2000

The second testbed was deployed on the second floor of a four-floor building, which had more than 30 rooms. Unlike the first testbed, five wall-mounted Wi-Fi access points were deployed. The layout for this testbed is visible on figure 4.2.

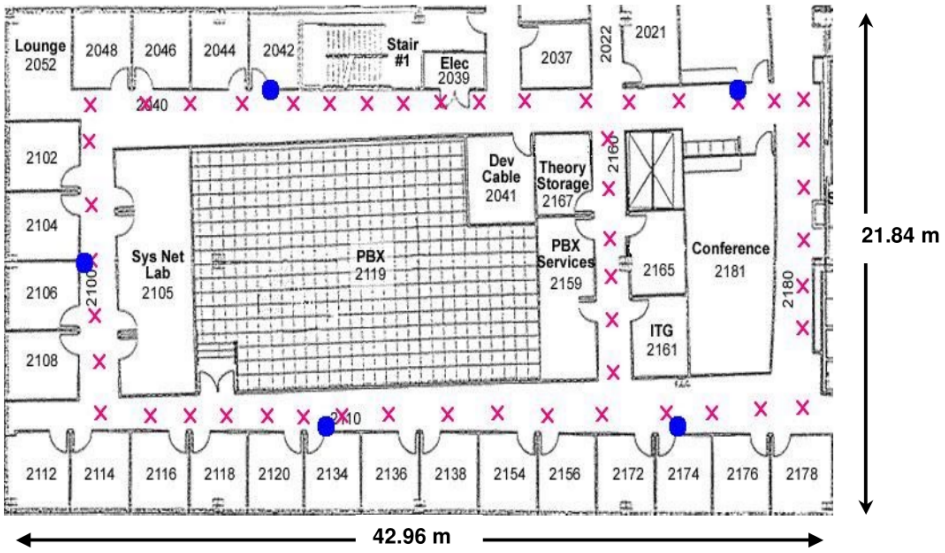

Figure 4.2: Floorplan of the site chosen for the first testbed. Circles represent the locations of the Wi-Fi APs, crosses the locations where signal strength samples were recorded. Image by Bahl, Paramvir and Padmanabhan, Venkata, Enhancements to the RADAR RF-based User Location and Tracking System, 2000

4.1.2 Data Collection

The RADAR algorithm requires information about the radio signal strength, since it assumes that it is a function of the user's location - visible on figure 4.3. This signal strength was first recorded at a testbed location during an off-line phase and is later used to infer a user's location in real time, during an on-line phase. In a sense, the off-line data set may be thought of as a *radio map*.

The mobile user's laptop was selected to be the source of Wi-Fi beacons, that were periodically broadcast. The APs then observed these beacons and extracted the signal strength (expressed in dBm) for each one of them, and recorded this information.

The broadcast-and-recording was automated, and proceeded as follows. Firstly, the clocks on the APs and the laptop were synchronized to within the round-trip time of a Wi-Fi link - less than 5 ms. The laptop then starts broadcasting UDP packets spaced uniformly at a rate of 4 per second. Each AP records the signal strength along with a timestamp. This information is collected during both the on and offline phases. During the offline phase only, the user's location was recorded along with each timestamp, as well as the direction they were facing - North, South, East and West.

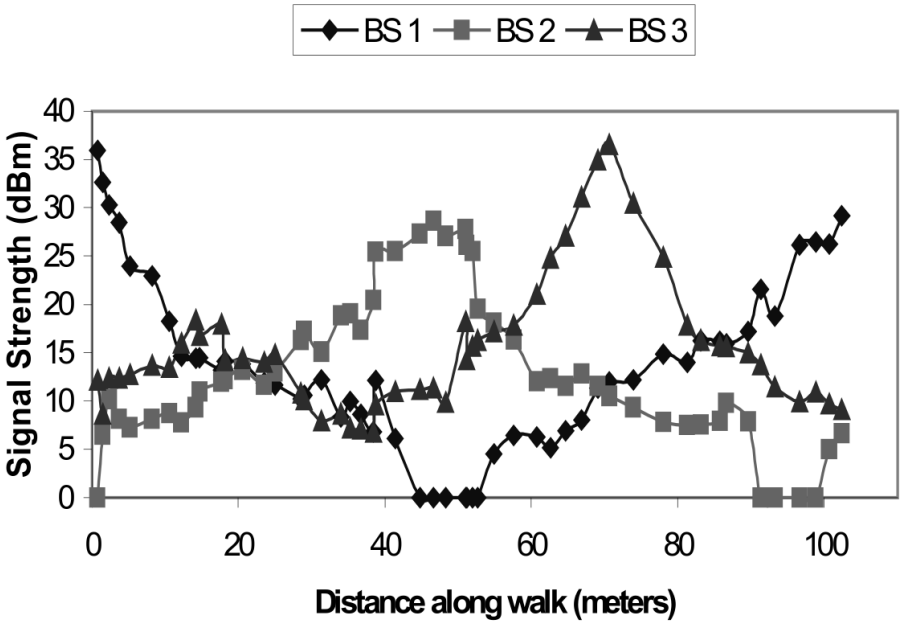

Figure 4.3: Variation of the signal strength recorded by three different APs observing a walking user. Image by Bahl, Paramvir and Padmanabhan, Venkata, RADAR: An In-Building RF-based User Location and Tracking System, 2000

For each combination of location and orientation, at least 20 signal strength samples were collected, for a total of 70 distinct physical locations.

4.1.3 Data Processing

By using the synchronized timestamps, the samples each AP collected were unified into a single table containing tuples of the form (x, y, d, ss_i, snr_i) where x and y represent the co-ordinates on the floorplan, d the direction the user was facing, ss_i and snr_i the signal strength and signal-to-noise ration respectively. The index i corresponds to each AP.

A simple linear-time search algorithm was employed to determine exact and closest matches.

4.1.4 Locating a user

One of RADAR's premises is that signal strength provides a way to infer a user's location, shown by observing the recorded signal strengths along a user's path and how they varied, as seen by each AP.

Given a set of signal strength measurements at each AP, the location that best matches the observed signal strength is determined, and represents a guess of the user's actual location.

The various signal strength samples for each base station were summarized by taking their mean.

Then, to determine the location and orientation that best matches a given summarized set of signal strength observations, the combination of location and single strength data gathered during the off-line phase is used. Although a signal propagation model was also used, it is not included here.

Finally a metric and search methodology are needed to compare multiple locations and select the one that best matches the observed signal strength. The technique originally used by RADAR was named *Nearest-neighbour in signal space (NNSS)*. The goal was to compute the distance in signal space between the observed set of signal strength measurements (ss_1, ss_2, ss_3) and the recorded measurements (ss'_1, ss'_2, ss'_3). The location that minimizes the distance is selected as the guess for the user's location. The distance function originally used was the *Euclidean Distance* function, $f(ss, ss') = \sum \sqrt{(ss_i - ss'_i)^2}$.

Basis Analysis Bahl and Padmanabhan picked one of the location-signal strength tuples in the search space at random and ran an NNSS search using the remainder of the tuples as a new search space, emulating locating a stationary user. They then compare this method - which they call the Empirical method - with two other methods, random selection and strongest base station selection. Random selection picks one of the points at random; strongest base station guesses the user's location to be the same as the location of the base station that recorded the strongest signal. Results of this analysis are visible on figure 4.4.

Multiple nearest neighbours Because the signal strength is highly variable with location, there may be multiple neighbours at roughly the same distance from the observed signal strengths. These neighbours may also be oriented in different directions. Because of this, multiple nearest neighbours may be selected and their positions averaged. The results obtained by Bahl and Padmanabhan indicate that averaging is beneficial but not significantly - for 5 nearest neighbours, the error distance between the guess and the real location was 22% better at the 25th percentile and 9% better for the 50th percentile. Accuracy degrades with a large number of neighbours because guesses that are very far from the user's true location are included in the average.

Method	25th (meter)	50th (meter)	75th (meter)
Empirical	1.92	2.94	4.69
Strongest	4.54 (2.4x)	8.16 (2.8x)	11.5 (2.5x)
Random	10.37 (5.4x)	16.26 (5.5x)	25.63 (5.5x)

Figure 4.4: The 25th, 50th and 75th percentile values for the error distance. In parenthesis are the numbers that represent the degradation compared to the Empirical method. Image by Bahl, Paramvir and Padmanabhan, Venkata, Image by Bahl, Paramvir and Padmanabhan, Venkata, RADAR: An In-Building RF-based User Location and Tracking System, 2000

Max Signal Strength Across Orientations Because orientation has an impact for estimating a location - consider the case where a user is occluding an AP - an analysis was performed in order to gauge how well RADAR would perform were orientation not an issue. For each location in the off-line data set, the maximum signal strength at each base station across all orientations was computed. This was done to emulate the case where the user's body is not obstructing the signal generated by the laptop. The results of this analysis are visible on figure 4.5.

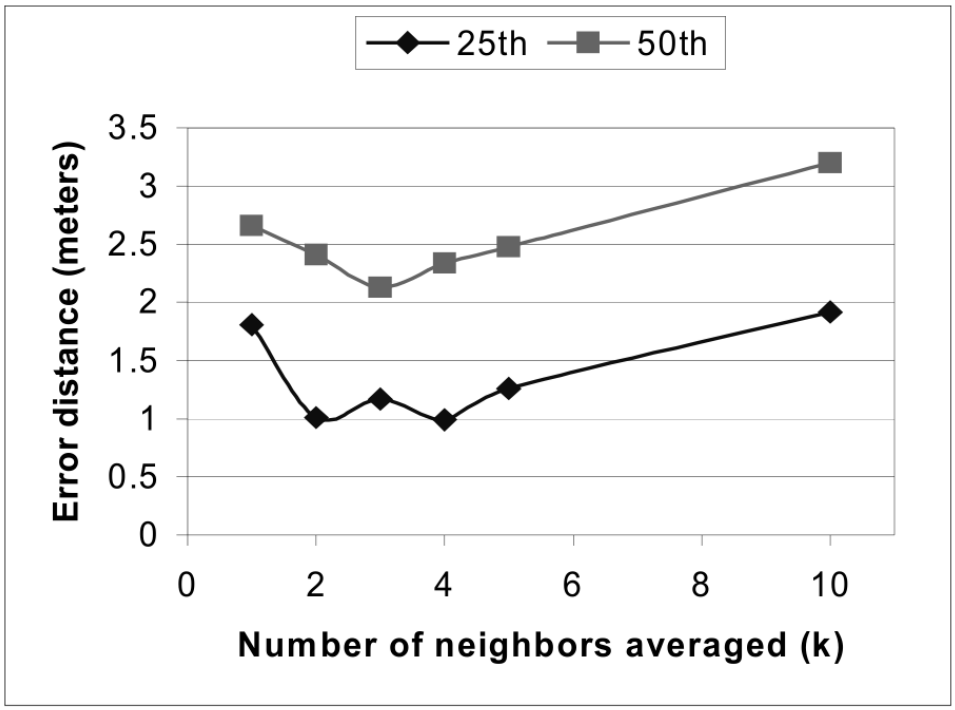

Figure 4.5: Error distance for the 25th and 50th percentile using the maximum signal strength across all orientations, for varying numbers of neighbours. Image by Bahl, Paramvir and Padmanabhan, Venkata, Image by Bahl, Paramvir and Padmanabhan, Venkata, RADAR: An In-Building RF-based User Location and Tracking System, 2000

Averaging over 2-4 neighbours increases RADAR's accuracy significantly.

Impact of Number of Data Points The impact of the number of physical locations on RADAR's accuracy was analysed. A random number of physical locations - ranging from 2 to 70 - was selected with which to build the NNSS search space. The results of this analysis are visible on figure 4.6.

RADAR performs almost as well with 40 locations as it would with the total 70, which means there is a lower bound for the system's accuracy.

Impact of the Number of Samples Because there may be a time constraint in collecting data from the user during the on-line phase, the impact of the number of samples that could be collected per location was analysed. The results obtained by Bahl and Padmanabhan show that with just 1 sample collected during the on-line phase the median error distance was 30% worse

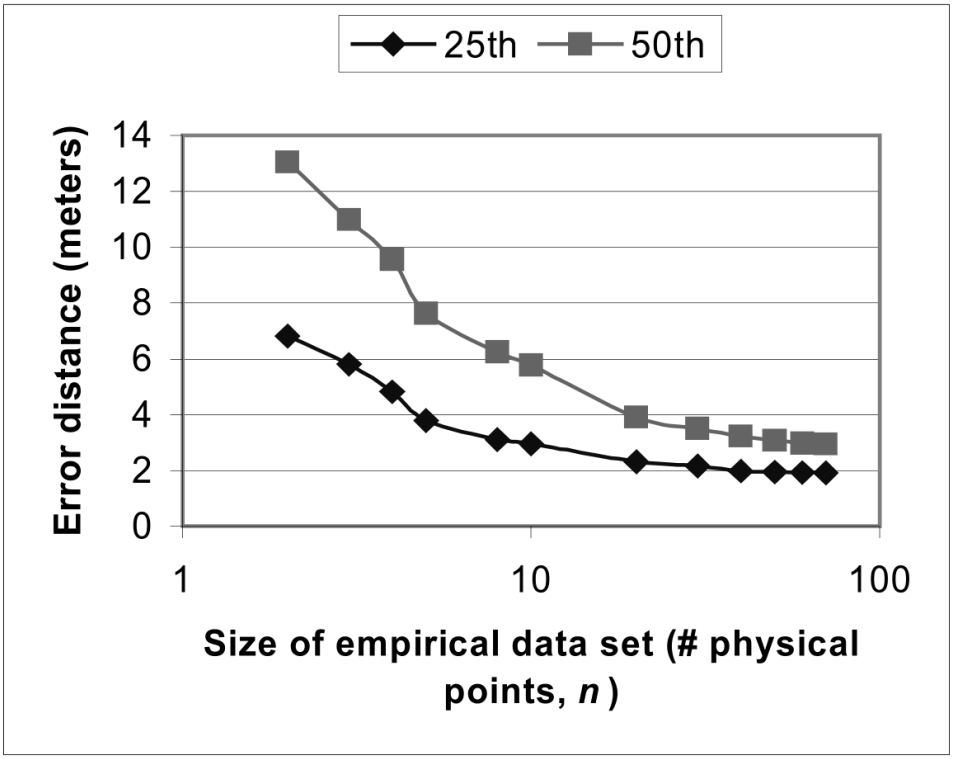

Figure 4.6: Error distance for the 25th and 50th percentile versus the size of the data set. Image by Bahl, Paramvir and Padmanabhan, Venkata, Image by Bahl, Paramvir and Padmanabhan, Venkata, RADAR: An In-Building RF-based User Location and Tracking System, 2000

than when all samples were collected, 11% worse with 2 and 4% worse with 3.

Impact of User Orientation The case where the off-line data set only has locations in a certain orientation, while observations correspond to the opposite orientation. The results obtained by Bahl and Padmanabhan show that the RADAR's accuracy is very degraded under these conditions - the 25th percentile of the error distance is 54% worse and the 50th percentile is 67% worse.

Tracking a Mobile User The problem of tracking a mobile user was tackled by having the APs record 4 signal strength samples per second. By using a sliding window of 10 samples to compute the mean signal strength on a continuous basis, the user's movements are tracked. It was observed that this method slightly degraded RADAR's accuracy, but enabled the continuous tracking of a user.

Because this method does not take into account the user's past positions, it was later improved

upon by using a "Viterbi-like" algorithm. Since an object cannot cross very large distances in an instant at random, the object's likely position should be close to an earlier position. The method used is as follows: initially, NNSS searches are performed to determine the k nearest neighbours. A history of h $k-$NNSS sets is kept in memory, indexed by time. These data are used to build a graph, illustrated on figure 4.7, with the NNSS as vertices and edges only between vertices that are contained in consecutive sets; the weight each edge is assigned is the euclidean physical distance between them. Each time a new observation is inserted into the history, the shortest path between the vertices in the oldest and newest sets is computed. The object's location is then guessed to be the point at the start of this path.

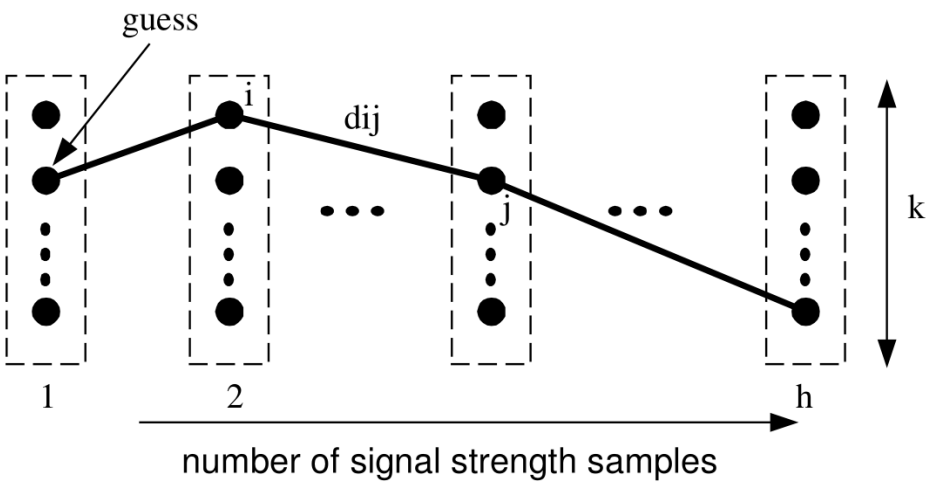

Figure 4.7: Depiction of the graph kept by the "Viterbi-like" algorithm. Shortest path is shown in bold. Weight of an edge between vertices i and j, d_{ij} is the Euclidean distance between the locations. Image by Bahl, Paramvir and Padmanabhan, Venkata, Enhancements to the RADAR RF-based User Location and Tracking System, 2000

The performance gains by using this method are illustrated on figure 4.8.

This algorithm outperforms any of the other algorithms. Of note, the 90th percentile of error distance being significantly smaller indicates that the contiguity restriction prevents RADAR from emitting a location guess that would indicate that the user 'jumped' across large distances between observations.

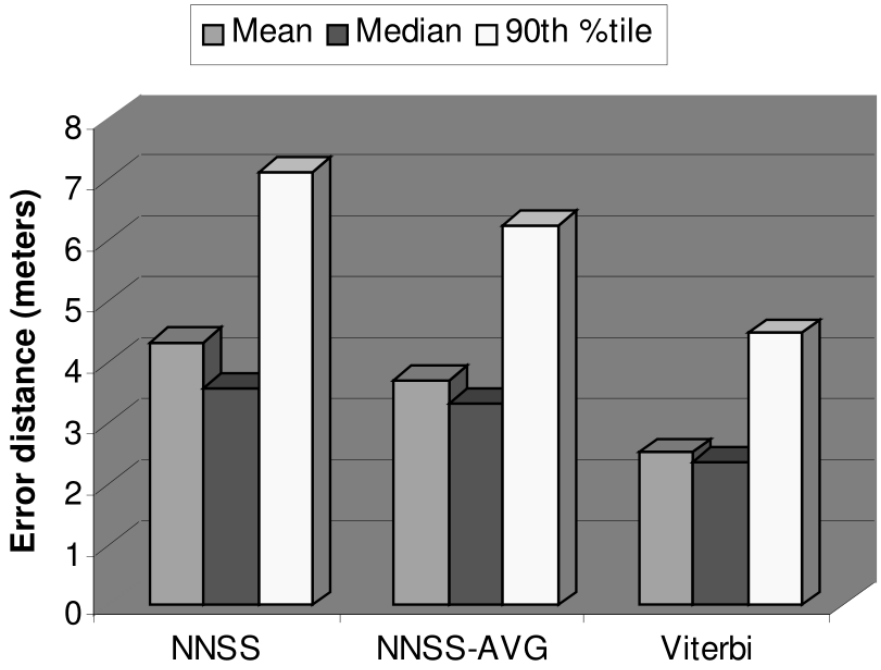

Figure 4.8: Performance of the Viterbi-like algorithm compared to NNSS and NNSS-Average. Image by Bahl, Paramvir and Padmanabhan, Venkata, Enhancements to the RADAR RF-based User Location and Tracking System, 2000

4.2 Adapting RADAR

In this section we discuss the implementation of an adaptaion of the RADAR system, detailing the differences from the original where pertinent.

4.2.1 Overview

As originally stated, the APs were the devices performing the sensing of the mobile user. However, no intervention on any AP was possible. Therefore the problem faced here is the reverse of the original - the moving object is sensing the APs. Although the original article noted little asymmetry, this fact required adapting the original algorithm for this reality.

Because the algorithm is resource intensive, the impact of the algorithm on a smartphone's resources was deemed too high. In addition, the sole database provided by the Android operating system - a SQLite implementation - was deemed too limited for this purpose, in addition to burdening the smartphone with a large volume of data. Therefore, the algorithm was split between

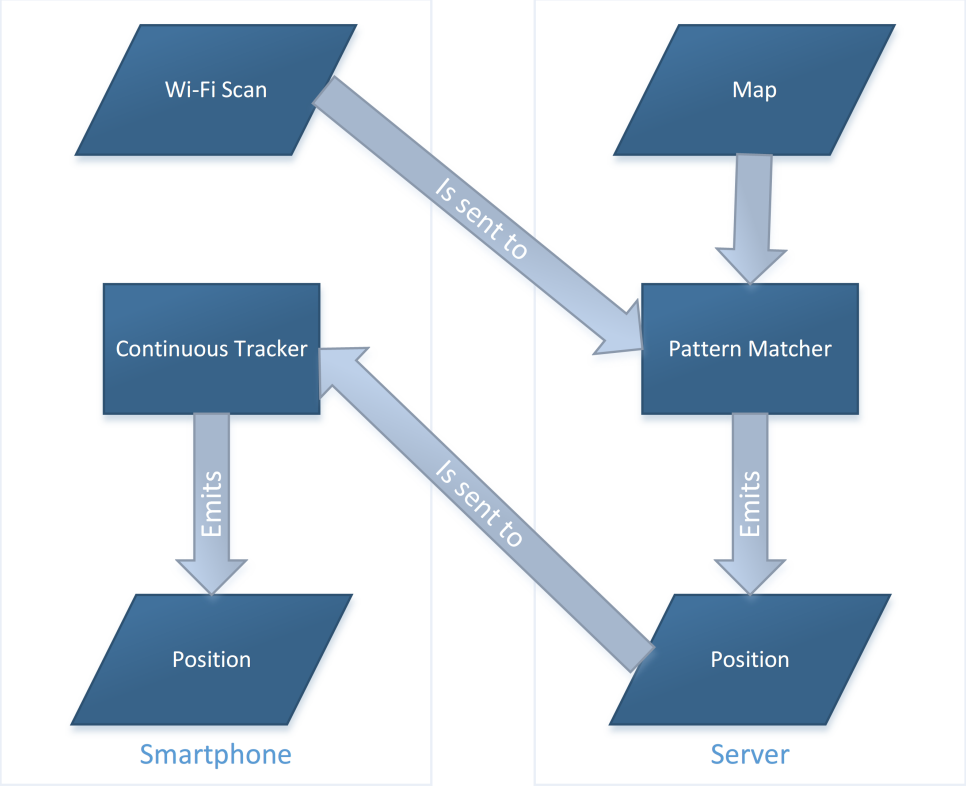

Figure 4.9: The implementation of the RADAR algorithm was split between a server, which hosted the map and performed the pattern matching, and the smartphone, which performed the user tracking.

a smartphone and a desktop computer - the smartphone senses the radio environment, sending this data to the computer via the internet; the computer then performs the pattern matching, comparing the received data from the smartphone with its map, and issuing a prediction about the user's likely location. Because the pattern matcher is stateless, the computer's job is ended, and the predictions are sent back to the smartphone, which keeps information about the user's past locations. Using the most current prediction, it performs the last step in the algorithm - a computation of the user's most likely path. An overview of the layout of the algorithm is visible in figure 4.9.

4.2.2 Architecture

4.2.2.1 Smartphone

The smartphone selected for experimentation and testing was a Samsung I9300 Galaxy S3, running the Android 4.3 operating system. The Android application targeted the Android API

level 21, equivalent to Android 4.1. The programming language used was Java.

4.2.2.2 Desktop computer

The desktop runs a Java Enterprise Edition application, designed to be stateless and provide quick and easy access to a database via HyperText Transfer Protocol (HTTP). This application runs on a GlassFish 4.1.0 application server.

The database is provided by a PostGIS 2.1.7 DataBase Management System (DBMS)

4.2.2.3 Communication

Communication between the smartphone and the desktop computer is carried through the Internet, using HTTP containing a JavaScript Object Notation (JSON) message.

4.2.3 Creating the Radio Map

The location where the map was built is the second floor of the Computer Science Department, University of Porto, the partial layout of which is detailed in figure 4.10. This site was selected for its low foot traffic, and low disruption and impact on the normal functioning of the building.

4.2.3.1 Sampling the environment

The sampling sites are separated by either 150 or 300 centimetres, and were taken in the centre of a corridor, as shown in figure 4.10.

At each site, 84 Wi-Fi scans - that is, going through the entire Wi-Fi gamut of frequencies and checking which APs were visible [18] - were performed, 21 per cardinal direction, and their results stored. Each scan yielded [19] a list of BSSID, along with their signal strength, our variable of interest. These results were coupled with information about the physical location at which they were collected, the orientation - North, South, East or West, and the time at which they were collected. These scans can therefore be thought of as representing a set of sets $S = \{\{BSSID, RSSI, location, orientation, timestamp\}, ...\}$.

In total, 1428 scans were performed.

Figure 4.10: Floorplan of the site where the radio environment was sampled. In red are the sites where signal strength samples were collected.

4.2.3.2 Storing the Map

With the above information, the map was created. Stored in a PostGIS database, the map is represented by a single table that stored each of the sets mentioned in 4.2.3.1.

Averaging the signals Because RADAR mostly makes use of the average signal strength, a second table was created that aggregated the average signal per location per BSSID and orientation.

```
      Column       |         Type          | Modifiers
-------------------+-----------------------+-----------
point              | text                  |
bssid              | character varying(17) |
orientation        | character(5)          |
```

```
avg          | numeric              |
stdev        | numeric              |
```

An example of a single entry:

```
POINT(0 0) | aa:bb:cc:dd:ee:ff | East | -79.57142857 | 1.69030850
```

The use of this table raised an immediate concern: because in most of the locations, some of the APs were visible in one instant, and invisible the next, how to best avoid the undue weight given the less often visible ones?

The chosen solution was to insert 'fake' readings - for each location and BSSID, the lowest signal strength was noted; every time this BSSID failed to appear at that location, a new entry was created that had the noted signal strength, with 1 dBm taken from it. This was done for each location and BSSID.

4.2.3.3 Using the map

In order to guess the object's location, the algorithm takes as its input a Wi-Fi scan from the object - the observation - and the radio map. The algorithm then proceeds to calculate the distance from the observation to all the locations on the map, and returns a set of possible locations, ordered by distance.

Distance functions We evaluated both Euclidean and Manhattan distances for comparison purposes. In either case, we define a few basic premisses for both distance functions, as follows:

- The distance between two different BSSIDs is infinite.
- It is assumed that new APs are not placed in the environment.
- Because map locations may contain BSSIDs that are not contained in an observation, a similar procedure to that in 4.2.3.2 is performed - that is, for each point considered, BSSIDs that are not contained in the observation are considered "just outside the limit of visible" and are added to it with their RSSI set to the lowest value on that point being considered, with 1 dBm subtracted.

- The same procedure is made in the case that an observation contains BSSIDs that do not appear in a certain map location.

Given a set of sets $M = \{\{BSSID, RSSI, location, orientation, timestamp\}, ...\}$ representing the map, and a set of sets $O = \{\{BSSID, RSSI, timestamp\}, ...\}$ representing the observations, distance calculations proceeds as follows: for every location in M all BSSIDs and their corresponding RSSIs are retrieved - in essence, the entire map is retrieved. These BSSID-RSSI pairs are then compared to the ones in the set O according to the basic premisses laid out - only BSSIDs that are equal on both sets are comparable. The RSSI values for each comparable BSSID on set M and set O are stored. The process continues until every location in M is compared to the set O.

Afterwards, the stored RSSI values are used in one of two possible distance functions: Euclidean and Manhattan distance. The Manhattan distance function is $f(d,p) = \sum |d_i - p_i|$, and the Euclidean distance function is $f(d,p) = \sum \sqrt{(d_i - p_i)^2}$, with d_i representing a RSSI for a particular BSSID on the map and p_i the same for an observation.

Finally, map locations are sorted by distance, and returned.

4.2.3.4 User tracking

User tracking was implemented as suggested by the original article, explained in 4.1.4. The Floyd-Warshall algorithm was used to find the all-pairs shortest path.

4.2.3.5 Experimental results

User standing still, no tracking A user holding the smartphone was placed facing North-NorthEast over each map location, and 21 scans were taken. The guesses provided by the algorithm were then compared to the actual physical locations the user was in, using Euclidean Distance to judge, without using user tracking. The results of this experiment are illustrated in figure 4.11.

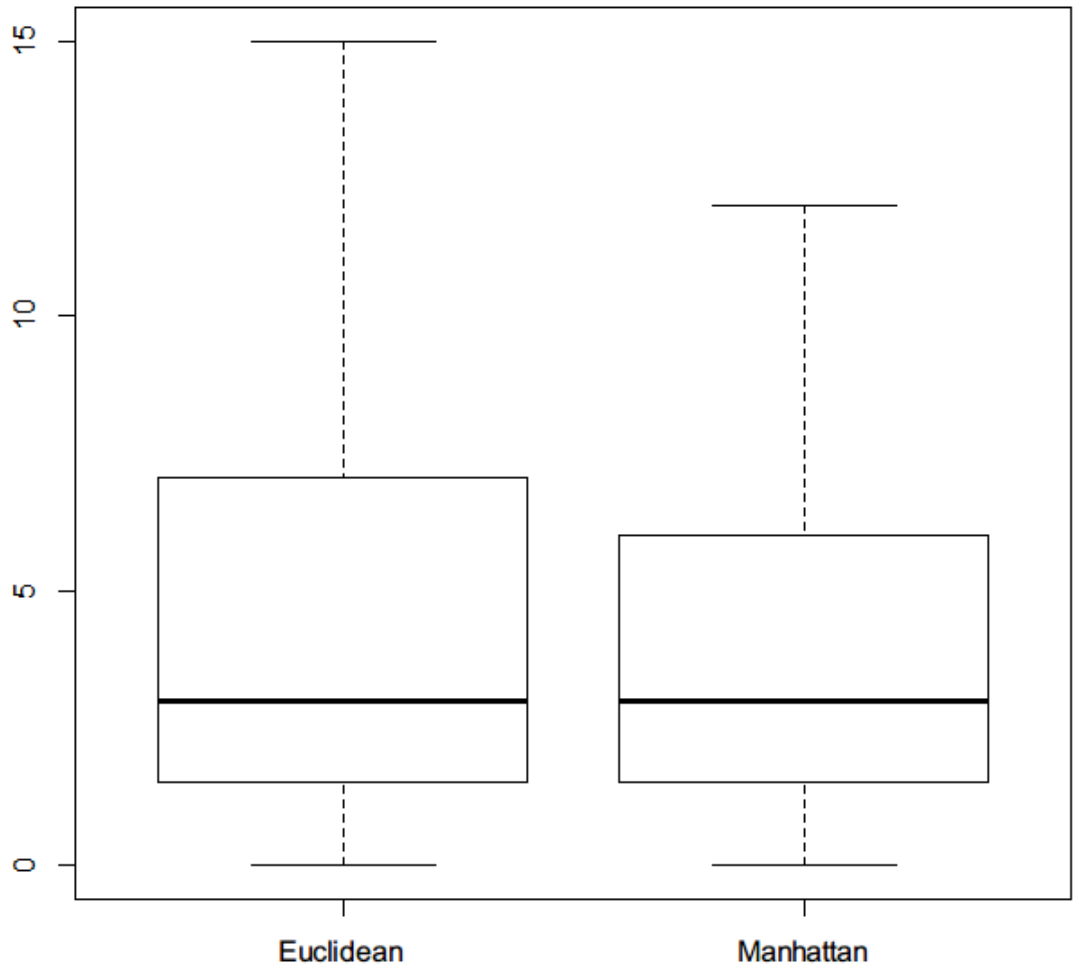

Figure 4.11: Box plot of the distances from the guesses emitted by the algorithm to the actual location, with a fixed user. Distances in metres.

User standing still, with tracking The same experiment was repeated, this with the user tracking mechanism enabled. The results are illustrated in figure 4.12

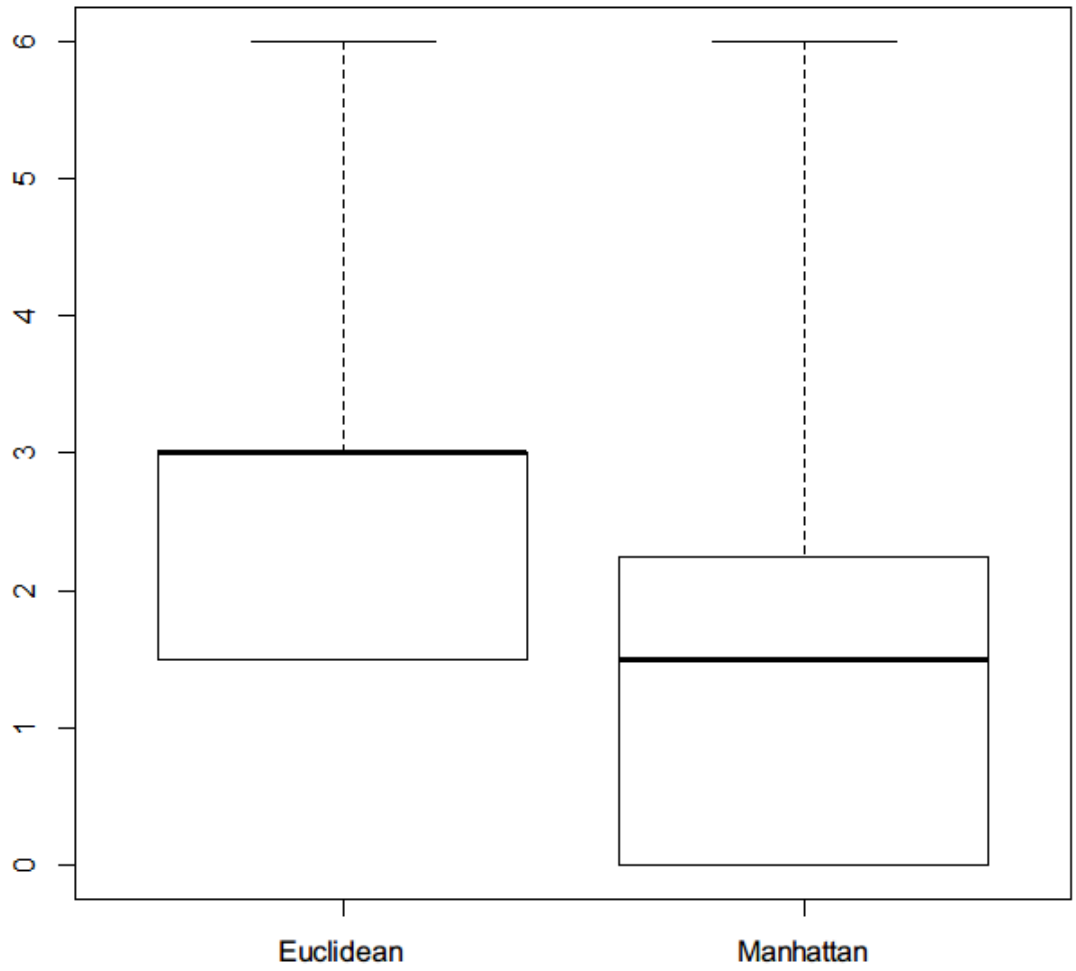

Figure 4.12: Box plot of the distances from the guesses emitted by the algorithm to the actual location, with a fixed user. Using the user tracking mechanism. Distances in metres.

User moving, with tracking In this experiment, the user walked along the corridor - from the perspective of figure 4.10, bottom to top. The experiment was repeated three times for each distance function. The results are illustrated in figure 4.13.

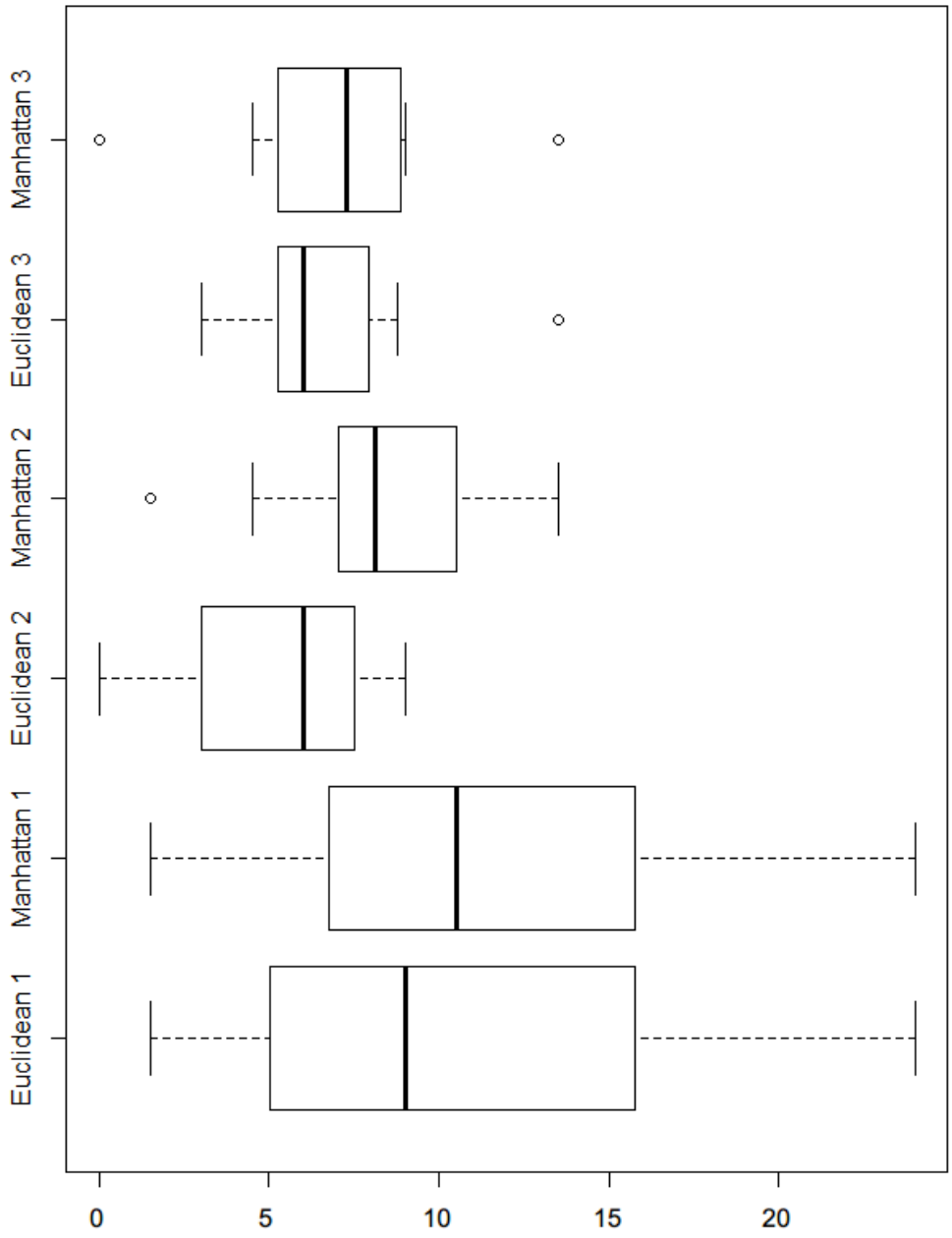

Figure 4.13: Box plot of the distances from the guesses emitted by the algorithm to the actual location, with a moving user. Using the user tracking mechanism. Distances in metres.

4.2.4 Discussion

In both experiments, the Manhattan distance function outperforms Euclidean distance - it more accurately guesses the user's position. The addition of the user tracking mechanism improves the performance of either distance function, giving us a median error distance in the case of Manhattan distance of 1.5 metres, and an average of 1.625 metres.

However, the moving user scenario severely affects the user tracking algorithm's performance. All medians are above 5 metres, and the guessed distances vary wildly with the experiment. We posit that the reason for this is the time which Wi-fi scans take - the device used needs about 4 seconds per scan, 16 times slower than the rate detailed in section 4.1.2. In essence, a scan starts while the user is at one location and ends while she is at another; the sampling of the radio environment is done in such a way that generates BSSID-RSSI pairs belonging to multiple locations that are not very near one another along a route.

Preliminary testing using other devices yielded scan times that were lower, indicating that the issue might be mitigated or totally solved by using those devices instead.

4.3 Summary

In this chapter we described the implementation details of the RADAR algorithm, changes made to the original, and experimental results.

Chapter 5

Conclusion and future work

This project intended to study the feasibility of using a smartphone and software in order to accurately guess a user's position indoors. To this end, two separate positioning systems were developed - one using inertial sensors and one using radio fingerprinting. In support of the radio fingerprinting system, a separate computer was used to provide computational power that a smartphone lacks, and the communication between these was established. These solutions were then successfully deployed and tested, proving its feasibility for commonplace use.

Lessons were learned regarding the fragmentation of the Android "eco-system": the very wide variety of hardware and software makes development and testing harder, due to certain peculiarities observed in certain hardware and software combinations - the fact that the selected testbed for the fingerprinting system had such a long time between Wi-Fi scans probably undermined the results from the fingerprinting system.

Unfortunately, and this work's biggest weakness, it was not possible, due to time constraints, to successfully fuse both solutions, into one that used both positioning methods to improve on the other's flaws.

There is much work to be done: a smarter and more efficient way to represent the radio map may improve the pattern matching performance, which was too slow to be used on a smartphone. An inertial-fingerprinting fusion method needs to be found, possibly including information from other radio sensors, such as smart watches that are paired with the smartphone. If this fusion proves successful, it may be possible to fully unconstrain the smartphone, allowing it total freedom to be positioned and rotated in whichever way, which would be a boon for blind users.

Finally, a GNSS solution may be integrated with this, in order to train the step length of the user - the system may be trained to know the user's characteristics, storing it for future use. This could be extended to allow seamless indoor and outdoor use.

Bibliography

[1] RFID Journal. Companies Deliver New Apps for Bluetooth Beacons - RFID Journal. http://www.rfidjournal.com/articles/view?11053, 2014. Consulted October 2014.

[2] Changdon Kee, Doohee Yun, Haeyoung Jun, Bradford Parkinson, Sam Pullen, and Tom Lagenstein. Centimeter-Accuracy Indoor Navigation Using GPS-Like Pseudolites. 2001.

[3] Paramvir Bahl and Venkata N Padmanabhan. RADAR: An In-Building RF-based User Location and Tracking System. 2000.

[4] Paramvir Bahl, Venkata N Padmanabhan, and Anand Balachandran. Enhancements to the RADAR user location and tracking system. 2000.

[5] Paul Castro, Patrick Chiu, Ted Kremenek, and Richard Muntz. A probabilistic room location service for wireless networked environments. pages 18–34, 2001.

[6] Thomas King, Stephan Kopf, Thomas Haenselmann, Christian Lubberger, and Wolfgang Effelsberg. Compass: A probabilistic indoor positioning system based on 802.11 and digital compasses. pages 34–40, 2006.

[7] Our Mobile Planet. Our mobile planet. https://think.withgoogle.com/mobileplanet/en/, 2015.

[8] Department of ECE - Capstone Projects. Capstone Project, 2014. http://users.encs.concordia.ca/~eceweb/capstone/projectshow.php?id=232.

[9] Play.google.com. Indoor Positioning. https://play.google.com/store/apps/details?id=com.bombao.projetwifi, 2015. Consulted December 2014.

[10] Wifarer.com. Indoor GPS technology, 2014. Consulted October 2014.

[11] Christian Henke. Smartnavi - step navigation. https://play.google.com/store/apps/details?id=com.ilm.sandwich&hl=en, 2015.

[12] www.idc.com. Idc: Smartphone os market share. https://www.idc.com/prodserv/smartphone-os-market-share.jsp, 2015.

[13] Google. Sensors overview. https://developer.android.com/guide/topics/sensors/sensors_overview.html, 2015.

[14] Google. Sensorevent. https://developer.android.com/reference/android/hardware/SensorEvent.html, 2015.

[15] Google. Sensormanager - rotation matrix. https://developer.android.com/reference/android/hardware/SensorManager.html#getRotationMatrix(float[],float[],float[],float[]), 2015.

[16] Neil Zhao. Full-featured pedometer design realized with 3-axis digital accelerometer. *Analogue Dialogue 44-06*, page 5, 2010.

[17] Paul Doust. What's the best 3d angular co-ordinate system for working with smartphone apps. https://math.stackexchange.com/questions/381649/whats-the-best-3d-angular-co-ordinate-system-for-working-with-smartphone-apps/382048#382048, 2015.

[18] Developer.android.com. Wifimanager, 2015.

[19] Developer.android.com. Scanresult, 2015.

Appendix A

Acronyms

AGPS Assisted GPS

ADOA Angle Difference of Arrival

AOA Angle of Arrival

AP Access Point

API Application Programming Interface

BSSID Basic service set identification

CPU Central processing unit

DBMS DataBase Management System

DR Dead Reckoning

GNSS Global Navigation Satellite Systems

GLONASS Globalnaya Navigatsionnaya Sputnikovaya Sistema

GPS Global Positioning System

HTTP HyperText Transfer Protocol

IEEE Institute of Electrical and Electronic Engineers

INS Inertial Navigation System

IR Infra-Red

ISM Industrial, Scientific and Medical radio band

JSON JavaScript Object Notation

LAN Local-Area Network

LOS Line-Of-Sight

MEMS Micro-electromechanical systems

NNSS Nearest-neighbour in signal space

NNSS-AVG Nearest-neighbour in signal space, averaged

LPD433 Low-powered device 433MHz

PAN Personal-Area Network

RF Radio-Frequency

RFID Radio-Frequency Identification

RSSI Received Signal Strength Indicator

SNR Signal-to-noise ratio

SSID Service set identification

TDOA Time Difference of Arrival

TOA Time of Arrival